

Введение в искусственный интеллект.

Машинное обучение

Лекция 9. Бустинг

Бабин Д.Н., Иванов И.Е., Петюшко А.А.

кафедра Математической Теории Интеллектуальных Систем

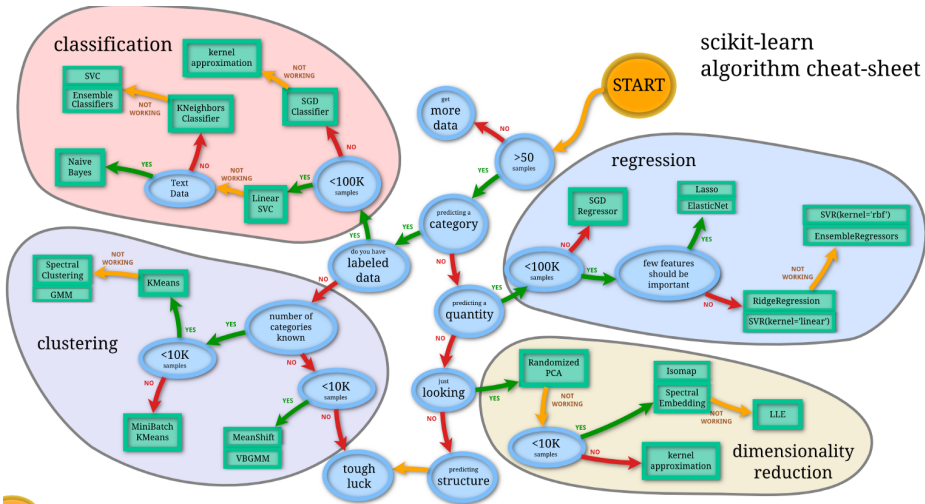
14 апреля 2020 г.



- 1 AdaBoost
- 2 AnyBoost
- 3 GB / SGB
- 4 TreeBoost

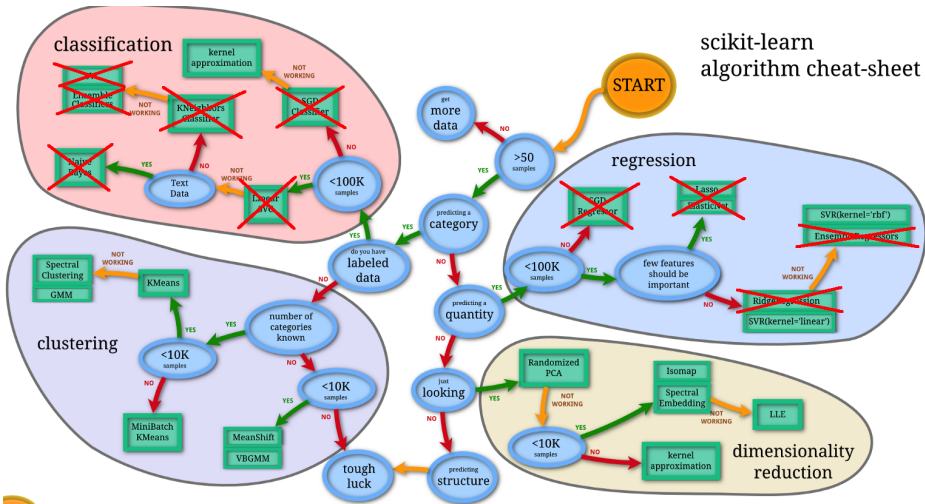


Дорожная карта Scikit-Learn¹



¹https://scikit-learn.org/stable/tutorial/machine_learning_map/

Дорожная карта Scikit-Learn¹



¹https://scikit-learn.org/stable/tutorial/machine_learning_map/

Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как

$$a(x) = \sum_{t=1}^T \alpha_t b_t, \alpha_t \in \mathbb{R}.$$


Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^T \alpha_t b_t$, $\alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$



Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^T \alpha_t b_t$, $\alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$

AnyBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из \mathbb{R} ,
- Функция потерь – гладкая функция от отступа $L(y_i a(x_i))$



Обозначим взвешенную сумму выходов базовых классификаторов $b_t(x)$ как $a(x) = \sum_{t=1}^T \alpha_t b_t, \alpha_t \in \mathbb{R}$.

AdaBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из дискретного множества (например, $\{-1, +1\}$),
- Функция потерь: $e^{-y_i a(x_i)}$

AnyBoost

- Базовые алгоритмы $b_t(x)$ принимают значения из \mathbb{R} ,
- Функция потерь – гладкая функция от отступа $L(y_i a(x_i))$

Gradient Boosting

- Базовые алгоритмы $b_t(x)$ принимают значения из \mathbb{R} ,
- Функция потерь – гладкая функция от пары $L(y_i, a(x_i))$



- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$



Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$



Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, 0 \leq \tilde{w}_i \leq 1$



Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, 0 \leq \tilde{w}_i \leq 1$
- Вероятностный вектор $U^m = (u_1, \dots, u_m)$: $\sum_{i=1}^m u_i = 1, u_i \geq 0$,



Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, 0 \leq \tilde{w}_i \leq 1$
- Вероятностный вектор $U^m = (u_1, \dots, u_m)$: $\sum_{i=1}^m u_i = 1, u_i \geq 0$,
- Взвешенное число правильных классификаций алгоритма $b(x)$ по вектору U^m :
 $P(b; U^m) = \sum_{i=1}^m u_i [b(x) = y_i]$
- Взвешенное число ошибочных классификаций алгоритма $b(x)$ по вектору U^m :
 $N(b; U^m) = \sum_{i=1}^m u_i [b(x) \neq y_i]$



Обозначения для AdaBoost

- Базовый алгоритм $b_t : X \rightarrow \{-1, +1\}$
- Вектор весов (взвешиваем объекты) $W^m = (w_1, \dots, w_m)$: $w_i = e^{-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i)}$
- Нормировка: $\tilde{w}_i = \frac{w_i}{\sum_{j=1}^m w_j} \Rightarrow \sum_{i=1}^m \tilde{w}_i = 1, 0 \leq \tilde{w}_i \leq 1$
- Вероятностный вектор $U^m = (u_1, \dots, u_m)$: $\sum_{i=1}^m u_i = 1, u_i \geq 0$,
- Взвешенное число правильных классификаций алгоритма $b(x)$ по вектору U^m :
 $P(b; U^m) = \sum_{i=1}^m u_i [b(x) = y_i]$
- Взвешенное число ошибочных классификаций алгоритма $b(x)$ по вектору U^m :
 $N(b; U^m) = \sum_{i=1}^m u_i [b(x) \neq y_i]$
- $P + N = 1$.



Пусть A – достаточно богатое семейство базовых алгоритмов.

Теорема

Если для любого нормированного вектора U^m существует алгоритм $b \in A$, т.ч. $N(b; U^m) < \frac{1}{2}$, то минимум аппроксимированного Э.Р. \tilde{R}_T достигается на:



Пусть A – достаточно богатое семейство базовых алгоритмов.

Теорема

Если для любого нормированного вектора U^m существует алгоритм $b \in A$, т.ч. $N(b; U^m) < \frac{1}{2}$, то минимум аппроксимированного Э.Р. \tilde{R}_T достигается на:

- $b_T = \arg \min_{b \in A} N(b; \tilde{W}^m)$



Пусть A – достаточно богатое семейство базовых алгоритмов.

Теорема

Если для любого нормированного вектора U^m существует алгоритм $b \in A$, т.ч. $N(b; U^m) < \frac{1}{2}$, то минимум аппроксимированного Э.Р. \tilde{R}_T достигается на:

- $b_T = \arg \min_{b \in A} N(b; \tilde{W}^m)$
- $\alpha_T = \frac{1}{2} \ln \frac{1 - N(b; \tilde{W}^m)}{N(b; \tilde{W}^m)}$



Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \dots, m,$

²Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Алгоритм AdaBoost²

Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение базового алгоритма $b_t = \arg \min_{b \in A} N(b; \widetilde{W}^m),$

²Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Алгоритм AdaBoost²

Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение базового алгоритма $b_t = \arg \min_{b \in A} N(b; \widetilde{W}^m),$
- Вычисление нового веса $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)},$

²Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Алгоритм AdaBoost²

Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение базового алгоритма $b_t = \arg \min_{b \in A} N(b; \widetilde{W}^m),$
- Вычисление нового веса $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)},$
- Обновление весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \dots, m,$

²Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Алгоритм AdaBoost²

Алгоритм

- Инициализация весов: $w_i = \frac{1}{m}, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение базового алгоритма $b_t = \arg \min_{b \in A} N(b; \tilde{W}^m),$
- Вычисление нового веса $\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \tilde{W}^m)}{N(b_t; \tilde{W}^m)},$
- Обновление весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}, i = 1, \dots, m,$
- Перенормировка весов $w_i := \frac{w_i}{\sum_{j=1}^m w_j}, i = 1, \dots, m.$

²Freund Y. and Schapire R.E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting"

Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$



Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$



Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$

- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m , то

$$\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)} > \frac{1}{2} \ln \frac{1}{2} = \frac{1}{2} \ln 1 = 0$$



Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m , то
$$\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)} > \frac{1}{2} \ln \frac{1}{2} = \frac{1}{2} \ln 1 = 0$$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,



Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m , то
$$\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)} > \frac{1}{2} \ln \frac{1}{2} = \frac{1}{2} \ln 1 = 0$$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,
- Вес объекта x_i уменьшается в e^{α_t} раз, когда b_t правильно его классифицирует,



Замечание относительно шага обновления весов $w_i := w_i e^{-\alpha_t y_i b_t(x_i)}$, $i = 1, \dots, m$.

- b_t ошибается на объекте $x_i \Rightarrow y_i \neq b_t(x_i) \Rightarrow y_i b_t(x_i) = -1$
- b_t правильно классифицирует объект $x_i \Rightarrow y_i = b_t(x_i) \Rightarrow y_i b_t(x_i) = +1$
- Поскольку $N(b; U^m) < \frac{1}{2}$ для любого нормированного U^m , то
$$\alpha_t = \frac{1}{2} \ln \frac{1 - N(b_t; \widetilde{W}^m)}{N(b_t; \widetilde{W}^m)} > \frac{1}{2} \ln \frac{1}{2} = \frac{1}{2} \ln 1 = 0$$
- Вес объекта x_i увеличивается в e^{α_t} раз, когда b_t допускает на нем ошибку,
- Вес объекта x_i уменьшается в e^{α_t} раз, когда b_t правильно его классифицирует,
- Т.о. наибольший вес будет у тех объектов, которые чаще неправильно классифицировались предыдущими алгоритмами (т.е. классификатору прежде всего нужно сосредоточиться именно на них!).



- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:



- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
 - Объекты с максимальными весами \tilde{w}_i , скорее всего, являются шумовыми выбросами



- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
 - Объекты с максимальными весами \tilde{w}_i , скорее всего, являются шумовыми выбросами
 - Их нужно исключить из выборки



- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
 - Объекты с максимальными весами \tilde{w}_i , скорее всего, являются шумовыми выбросами
 - Их нужно исключить из выборки
 - После чего начать построение композиции заново



- После построения некоторого количества базовых алгоритмов (например, $T = 10 \dots 30$) можно проанализировать распределение весов объектов:
 - Объекты с максимальными весами \tilde{w}_i , скорее всего, являются шумовыми выбросами
 - Их нужно исключить из выборки
 - После чего начать построение композиции заново
- Бустинг можно использовать как универсальный метод фильтрации выбросов перед применением любого другого метода классификации



Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции



Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
 - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!



Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
 - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Теоретическое обоснование “на пальцах”*:
взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов



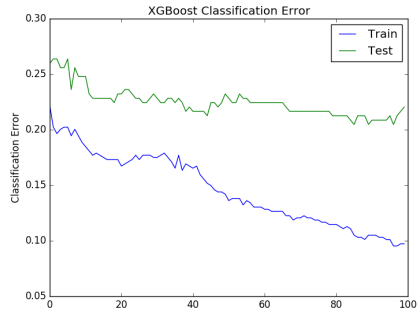
Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
 - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Теоретическое обоснование “на пальцах”*:
взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов
 - Т.к. стараемся увеличить отступы
$$y_i \sum_{t=1}^T \alpha_t b_t(x_i)$$




Обобщающая способность бустинга: эмпирические замечания

- Во многих экспериментах тестовая ошибка практически постоянно уменьшалась по мере увеличения числа алгоритмов в композиции
 - Часто тестовая ошибка уменьшалась даже после достижения нулевой ошибки на обучающей выборке!
- *Теоретическое обоснование “на пальцах”*: взвешенное голосование не увеличивает эффективную сложность алгоритма (т.о. не переобучаемся), а сглаживает ответы базовых алгоритмов
 - Т.к. стараемся увеличить отступы
$$y_i \sum_{t=1}^T \alpha_t b_t(x_i)$$
 - Тем не менее, бустинг не идеален: иногда получается его переобучить





Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,

³R. Schapire et al (1998). "Boosting the margin: A new explanation for the effectiveness of voting methods" 


Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,
- Ансамбль $a(x) = \sum_{t=1}^T \alpha_t b_t(x)$ (без sign!),

³R. Schapire et al (1998). "Boosting the margin: A new explanation for the effectiveness of voting methods"  

Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,
- Ансамбль $a(x) = \sum_{t=1}^T \alpha_t b_t(x)$ (без sign!),
- D – вероятностное распределение над $X \times \{-1, +1\}$,

³R. Schapire et al (1998). "Boosting the margin: A new explanation for the effectiveness of voting methods" 

Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,
- Ансамбль $a(x) = \sum_{t=1}^T \alpha_t b_t(x)$ (без sign!),
- D – вероятностное распределение над $X \times \{-1, +1\}$,
- S – m независимых примеров из D (например, множество X^m). Тогда верна

³R. Schapire et al (1998). “Boosting the margin: A new explanation for the effectiveness of voting methods”

Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,
- Ансамбль $a(x) = \sum_{t=1}^T \alpha_t b_t(x)$ (без sign!),
- D – вероятностное распределение над $X \times \{-1, +1\}$,
- S – m независимых примеров из D (например, множество X^m). Тогда верна

Теорема

Для $\forall \theta > 0$, $\forall 0 < \delta < 1$ с вероятностью $1 - \delta$ верно следующее:

$$\mathbf{P}_D(ya(x) \leq 0) \leq \mathbf{P}_S(ya(x) \leq \theta) + \mathbf{O} \left(\sqrt{\frac{\ln m \ln |A|}{m\theta^2}} + \frac{1}{m} \ln \frac{1}{\delta} \right)$$

³R. Schapire et al (1998). “Boosting the margin: A new explanation for the effectiveness of voting methods”

Обобщающая способность бустинга: теория³

- Пусть семейство базовых алгоритмов A конечно: $|A| < \infty$,
- Ансамбль $a(x) = \sum_{t=1}^T \alpha_t b_t(x)$ (без sign!),
- D – вероятностное распределение над $X \times \{-1, +1\}$,
- S – m независимых примеров из D (например, множество X^m). Тогда верна

Теорема

Для $\forall \theta > 0$, $\forall 0 < \delta < 1$ с вероятностью $1 - \delta$ верно следующее:

$$P_D(ya(x) \leq 0) \leq P_S(ya(x) \leq \theta) + O\left(\sqrt{\frac{\ln m \ln |A|}{m\theta^2}} + \frac{1}{m} \ln \frac{1}{\delta}\right)$$

Вывод. Верхняя оценка не зависит от T : с увеличением T растет значение $ya(x) \Rightarrow$ увеличивая θ , уменьшим верхнюю оценку и в итоге улучшим обобщающую способность.

³R. Schapire et al (1998). "Boosting the margin: A new explanation for the effectiveness of voting methods" 

- Что использовать в качестве базовых классификаторов:



- Что использовать в качестве базовых классификаторов:
 - Чаще всего используют решающие деревья



- Что использовать в качестве базовых классификаторов:
 - Чаще всего используют решающие деревья
 - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \leq r_j]$, где $x = (f_1(x), \dots, f_n(x)) \in \mathbb{R}^n$



- Что использовать в качестве базовых классификаторов:
 - Чаще всего используют решающие деревья
 - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \leq r_j]$, где $x = (f_1(x), \dots, f_n(x)) \in \mathbb{R}^n$
 - SVM используется редко (обучается достаточно долго, прироста большого не дает)



- Что использовать в качестве базовых классификаторов:
 - Чаще всего используют решающие деревья
 - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \leq r_j]$, где $x = (f_1(x), \dots, f_n(x)) \in \mathbb{R}^n$
 - SVM используется редко (обучается достаточно долго, прироста большого не дает)
- Если вдруг при обучении получается нулевая ошибка ($N = 0$), то формула для выбора оптимального коэффициента приобретает вид $\alpha = \frac{1}{2} \ln \frac{1 - N + \frac{1}{m}}{N + \frac{1}{m}} = \frac{1}{2} \ln(m + 1)$



- Что использовать в качестве базовых классификаторов:
 - Чаще всего используют решающие деревья
 - Также используют совсем вырожденные случаи – т.н. “пни”: $b(x) = [f_j(x) \leq r_j]$, где $x = (f_1(x), \dots, f_n(x)) \in \mathbb{R}^n$
 - SVM используется редко (обучается достаточно долго, прироста большого не дает)
- Если вдруг при обучении получается нулевая ошибка ($N = 0$), то формула для выбора оптимального коэффициента приобретает вид $\alpha = \frac{1}{2} \ln \frac{1 - N + \frac{1}{m}}{N + \frac{1}{m}} = \frac{1}{2} \ln(m + 1)$
- Нужно периодически производить фильтрацию выбросов в обучающей выборке



Визуализация работы основных методов классификации

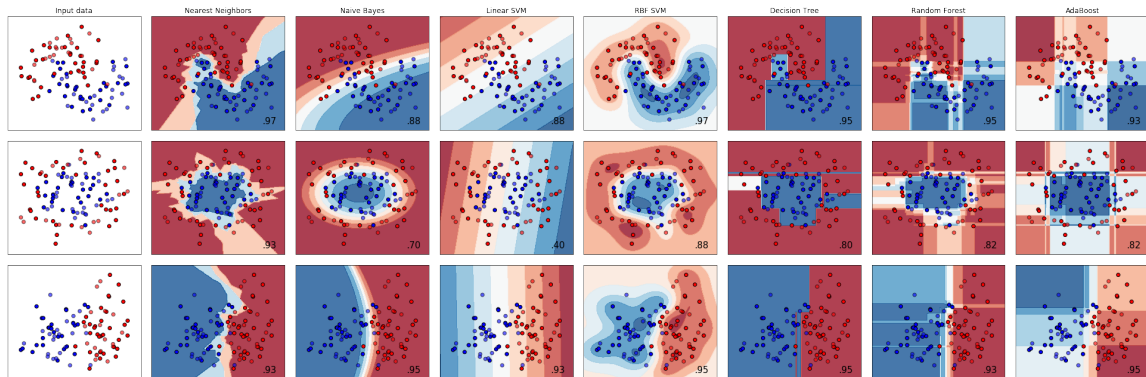
Посмотрим результаты работы основных классификаторов на трех разных задачах⁴.

⁴[https:](https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)

[//scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html](https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)

Визуализация работы основных методов классификации

Посмотрим результаты работы основных классификаторов на трех разных задачах⁴.



⁴https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема
- Базовые алгоритмы должны быть достаточно простыми, и их должно быть много (а лучше бы наоборот)



Плюсы и минусы AdaBoost

Плюсы

- Хорошая обобщающая способность (сложно переобучить)
- Простота реализации
- Время обучения ансамбля (веса) на порядок меньше времени обучения базовых алгоритмов
- Можно фильтровать выбросы

Минусы

- Чувствителен к выбросам
- Композиция совершенно неинтерпретируема
- Базовые алгоритмы должны быть достаточно простыми, и их должно быть много (а лучше бы наоборот)
- Необходимость в достаточно большой обучающей выборке (т.к. нет процедуры бутстрэпа)





Перейдём к более общему случаю:

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent” 



Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”  

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$



⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”  

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \rightarrow \min_{\alpha_T, b_T}.$$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”  

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \rightarrow \min_{\alpha_T, b_T}.$$

Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0 :

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0).$$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \rightarrow \min_{\alpha_T, b_T}.$$

Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0 :

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0).$$

Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)$, $x_0 = h_{T-1}(x_i)$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:

$$R_T \leq \tilde{R}_T = \sum_{i=1}^m L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \rightarrow \min_{\alpha_T, b_T}.$$

Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0 :

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0).$$

Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)$, $x_0 = h_{T-1}(x_i)$
- Тогда $\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) + \alpha_T \sum_{i=1}^m L'(h_{T-1}(x_i)) y_i b_T(x_i)$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). “Boosting algorithms as gradient descent”

Перейдём к более общему случаю:

- Недискретным ответам базовых алгоритмов, т.е. $b_t : X \rightarrow \mathbb{R}$
- Функции потерь $L(h_T)$, гладкой от отступа $h_T(x_i) = y_i \sum_{t=1}^T \alpha_t b_t(x_i)$

Принцип минимизации аппроксимированного Э.Р.:



$$R_T \leq \tilde{R}_T = \sum_{i=1}^m L(h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)) \rightarrow \min_{\alpha_T, b_T}.$$

Вспомним разложение Тейлора функции $f(x)$ в окрестности точки x_0 :

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0).$$

Воспользуемся этим разложением для аппроксимированного Э.Р.:

- Пусть $x = h_{T-1}(x_i) + \alpha_T y_i b_T(x_i)$, $x_0 = h_{T-1}(x_i)$
- Тогда $\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) + \alpha_T \sum_{i=1}^m L'(h_{T-1}(x_i)) y_i b_T(x_i)$
- Обозначив за $w_i = -L'(h_{T-1}(x_i))$, получаем
$$\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^m w_i y_i b_T(x_i)$$

⁵Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. R. (2000). "Boosting algorithms as gradient descent"  

$$\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^m w_i y_i b_T(x_i)$$

Зафиксировав α_T , переходим от задачи двумерной оптимизации $\tilde{R}_T \rightarrow \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):



$$\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^m w_i y_i b_T(x_i)$$

Зафиксировав α_T , переходим от задачи двумерной оптимизации $\tilde{R}_T \rightarrow \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):

$$\sum_{i=1}^m w_i y_i b_T(x_i) \rightarrow \max_{b_T}$$



$$\tilde{R}_T \approx \sum_{i=1}^m L(h_{T-1}(x_i)) - \alpha_T \sum_{i=1}^m w_i y_i b_T(x_i)$$

Зафиксировав α_T , переходим от задачи двумерной оптимизации $\tilde{R}_T \rightarrow \min_{\alpha_T, b_T}$ к одномерной (по алгоритму):

$$\sum_{i=1}^m w_i y_i b_T(x_i) \rightarrow \max_{b_T}$$

Затем определяем α_T , подставив найденный b_T .



Алгоритм

- Инициализация отступов: $h_0(x_i) = 0, i = 1, \dots, m,$



Алгоритм

- Инициализация отступов: $h_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h_{t-1}(x_i)),$



Алгоритм

- Инициализация отступов: $h_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h_{t-1}(x_i)),$
- Обучение нового базового алгоритма $b_t = \arg \max_{b \in A} \sum_{i=1}^m w_i y_i b(x_i),$



Алгоритм

- Инициализация отступов: $h_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h_{t-1}(x_i)),$
- Обучение нового базового алгоритма $b_t = \arg \max_{b \in A} \sum_{i=1}^m w_i y_i b(x_i),$
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i=1}^m L(h_{t-1}(x_i) + \alpha y_i b_t(x_i)),$



Алгоритм

- Инициализация отступов: $h_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Вычисление весов $w_i = -L'(h_{t-1}(x_i)),$
- Обучение нового базового алгоритма $b_t = \arg \max_{b \in A} \sum_{i=1}^m w_i y_i b(x_i),$
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i=1}^m L(h_{t-1}(x_i) + \alpha y_i b_t(x_i)),$
- Обновление отступов $h_t(x_i) = h_{t-1}(x_i) + \alpha_t y_i b_t(x_i).$



Градиентный бустинг⁶ – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.

Функционал качества: $\tilde{R}_T = \sum_{i=1}^m L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$.

⁶Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг⁶ – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.

Функционал качества: $\tilde{R}_T = \sum_{i=1}^m L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$.

Обозначения:

- Приближение для объекта x_i на шаге t : $f_t(x_i)$,

⁶Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг⁶ – обозначения

Рассмотрим самый общий случай – произвольную функцию потерь $L(a, y)$.

Функционал качества: $\tilde{R}_T = \sum_{i=1}^m L(\sum_{t=1}^{T-1} \alpha_t b_t(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$.

Обозначения:

- Приближение для объекта x_i на шаге t : $f_t(x_i)$,
- Тогда функционал качества примет вид:
$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

⁶Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$



Градиентный бустинг – обоснование

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение **градиентного спуска** для данной задачи:



Градиентный бустинг – обоснование

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение **градиентного спуска** для данной задачи:

- $f_T(x_i) = f_{T-1}(x_i) - \eta g_i$, где $g_i = L'(f_{T-1}(x_i), y_i)$



Градиентный бустинг – обоснование

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение **градиентного спуска** для данной задачи:

- $f_T(x_i) = f_{T-1}(x_i) - \eta g_i$, где $g_i = L'(f_{T-1}(x_i), y_i)$

Сравните с итерацией **бустинга**:

- $f_T(x_i) = f_{T-1}(x_i) + \alpha_T b_T(x_i)$



Градиентный бустинг – обоснование

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение **градиентного спуска** для данной задачи:

- $f_T(x_i) = f_{T-1}(x_i) - \eta g_i$, где $g_i = L'(f_{T-1}(x_i), y_i)$

Сравните с итерацией **бустинга**:

- $f_T(x_i) = f_{T-1}(x_i) + \alpha_T b_T(x_i)$



Градиентный бустинг – обоснование

$$\tilde{R}_T = \sum_{i=1}^m L(f_{T-1}(x_i) + \alpha_T b_T(x_i), y_i) \rightarrow \min_{\alpha_T, b_T}$$

Применение **градиентного спуска** для данной задачи:

- $f_T(x_i) = f_{T-1}(x_i) - \eta g_i$, где $g_i = L'(f_{T-1}(x_i), y_i)$

Сравните с итерацией **бустинга**:

- $f_T(x_i) = f_{T-1}(x_i) + \alpha_T b_T(x_i)$



Основная идея градиентного бустинга

Поиск нового базового алгоритма b_T для приближения антиградиента $(-L'(f_{T-1}(x_i), y_i))$, т.е. минимизация квадратичной ошибки:

$$b_T = \arg \min_{b \in A} \sum_{i=1}^m (b(x_i) - (-L'(f_{T-1}(x_i), y_i)))^2.$$

Алгоритм

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m,$



Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i=1}^m (b(x_i) + L'(f_{t-1}(x_i), y_i))^2,$



Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i=1}^m (b(x_i) + L'(f_{t-1}(x_i), y_i))^2,$
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i=1}^m L(f_{t-1}(x_i) + \alpha b_t(x_i), y_i),$



Алгоритм градиентного бустинга

Алгоритм

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m,$

Для $t = 1, \dots, T$

- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i=1}^m (b(x_i) + L'(f_{t-1}(x_i), y_i))^2,$
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i=1}^m L(f_{t-1}(x_i) + \alpha b_t(x_i), y_i),$
- Обновление приближений $f_t(x_i) = f_{t-1}(x_i) + \alpha_t b_t(x_i), i = 1, \dots, m.$



Связь градиентного бустинга с другими вариантами бустинга

На самом деле, градиентный бустинг – наиболее общий вариант бустинга:

Связь градиентного бустинга с другими вариантами бустинга

На самом деле, градиентный бустинг – наиболее общий вариант бустинга:

- Если положим $L(y_i, a(x_i)) = L(-y_i a(x_i))$, то получим AnyBoost,



Связь градиентного бустинга с другими вариантами бустинга

На самом деле, градиентный бустинг – наиболее общий вариант бустинга:

- Если положим $L(y_i, a(x_i)) = L(-y_i a(x_i))$, то получим AnyBoost,
- Если положим $L(y_i, a(x_i)) = e^{-y_i a(x_i)}$ и ограничим $b_t(x_i) \in \{-1, +1\}$, то получим AdaBoost.



Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m,$

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i \in I} (b(x_i) + L'(f_{t-1}(x_i), y_i))^2$,

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i \in I} (b(x_i) + L'(f_{t-1}(x_i), y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i \in I} L(f_{t-1}(x_i) + \alpha b_t(x_i), y_i)$,

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i \in I} (b(x_i) + L'(f_{t-1}(x_i), y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i \in I} L(f_{t-1}(x_i) + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_t(x_i) = f_{t-1}(x_i) + \alpha_t b_t(x_i), i \in I$.

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Стохастический градиентный бустинг

Используем не всю обучающую выборку, а случайное подмножество объектов.

Алгоритм SGB⁷

- Инициализация приближений: $f_0(x_i) = 0, i = 1, \dots, m$,

Для $t = 1, \dots, T$

- Выбор случайного подмножества $I \subseteq \{1, \dots, m\}$,
- Обучение нового базового алгоритма $b_t = \arg \min_{b \in A} \sum_{i \in I} (b(x_i) + L'(f_{t-1}(x_i), y_i))^2$,
- Вычисление нового веса $\alpha_t = \arg \min_{\alpha} \sum_{i \in I} L(f_{t-1}(x_i) + \alpha b_t(x_i), y_i)$,
- Обновление приближений $f_t(x_i) = f_{t-1}(x_i) + \alpha_t b_t(x_i), i \in I$.

Замечание. Последние два шага можно делать и для всей выборки, но это медленнее (хотя и точнее).

⁷Friedman, J. H. (1999). "Stochastic gradient boosting".

Плюсы SGB

- Уменьшение времени обучения



Плюсы SGB

- Уменьшение времени обучения
 - Меньше объектов на каждом шаге



Плюсы SGB

- Уменьшение времени обучения
 - Меньше объектов на каждом шаге
- Ускорение сходимости



Плюсы SGB

- Уменьшение времени обучения
 - Меньше объектов на каждом шаге
- Ускорение сходимости
 - Меньше шагов



Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J_t} r_j^t [x \in R_j^t]$, где

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J^t} r_j^t [x \in R_j^t]$, где

- Пространство делится на J^t непересекающихся областей (листов) $R_1^t, \dots, R_{J^t}^t$,

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J^t} r_j^t [x \in R_j^t]$, где

- Пространство делится на J^t непересекающихся областей (листов) $R_1^t, \dots, R_{J^t}^t$,
- Значение r_j^t в листе R_j^t – это среднее значение по обучающим примерам из этой области:
$$r_j^t = \frac{\sum_{i=1}^m y_i [x_i \in R_j^t]}{\sum_{i=1}^m [x_i \in R_j^t]}$$

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J^t} r_j^t [x \in R_j^t]$, где

- Пространство делится на J^t непересекающихся областей (листов) $R_1^t, \dots, R_{J^t}^t$,
- Значение r_j^t в листе R_j^t – это среднее значение по обучающим примерам из этой области:
$$r_j^t = \frac{\sum_{i=1}^m y_i [x_i \in R_j^t]}{\sum_{i=1}^m [x_i \in R_j^t]}$$

Варианты бустинга на деревьях⁸:

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J^t} r_j^t [x \in R_j^t]$, где

- Пространство делится на J^t непересекающихся областей (листов) $R_1^t, \dots, R_{J^t}^t$,
- Значение r_j^t в листе R_j^t – это среднее значение по обучающим примерам из этой области: $r_j^t = \frac{\sum_{i=1}^m y_i [x_i \in R_j^t]}{\sum_{i=1}^m [x_i \in R_j^t]}$

Варианты бустинга на деревьях⁸:

- **Общего вида.** На каждом шаге находится ровно один параметр α_t :
$$f_t(x_i) = f_{t-1}(x_i) + \alpha_t b_t(x_i),$$

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Градиентный бустинг на решающих деревьях

Пусть каждый базовый алгоритм – это CART-дерево $b_t(x) = \sum_{j=1}^{J^t} r_j^t [x \in R_j^t]$, где

- Пространство делится на J^t непересекающихся областей (листов) $R_1^t, \dots, R_{J^t}^t$,
- Значение r_j^t в листе R_j^t – это среднее значение по обучающим примерам из этой области: $r_j^t = \frac{\sum_{i=1}^m y_i [x_i \in R_j^t]}{\sum_{i=1}^m [x_i \in R_j^t]}$

Варианты бустинга на деревьях⁸:

- **Общего вида.** На каждом шаге находится ровно один параметр α_t :
$$f_t(x_i) = f_{t-1}(x_i) + \alpha_t b_t(x_i),$$
- **Улучшенный.** На каждом шаге находятся J^t параметров α_j^t :
$$f_t(x_i) = f_{t-1}(x_i) + \sum_{j=1}^{J^t} \alpha_j^t r_j^t [x_i \in R_j^t].$$

⁸Friedman, J. H. (1999). "Greedy function approximation: a gradient boosting machine".

Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях⁹:

⁹<https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>

¹⁰Chen, T., and Guestrin, C. (2016). "Xgboost: A scalable tree boosting system".

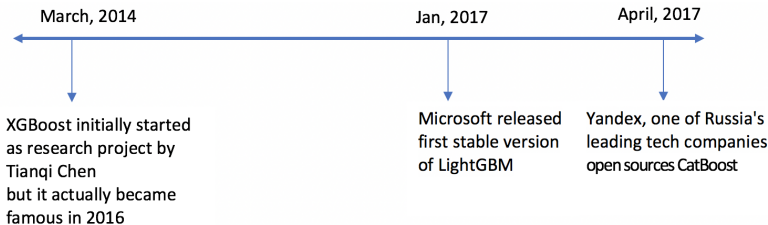
¹¹Ke, G. et al. (2017). "Lightgbm: A highly efficient gradient boosting decision tree".

¹²Prokhorenkova, L., et al. (2017). "CatBoost: unbiased boosting with categorical features".



Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях⁹:



⁹<https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>

¹⁰Chen, T., and Guestrin, C. (2016). "Xgboost: A scalable tree boosting system".

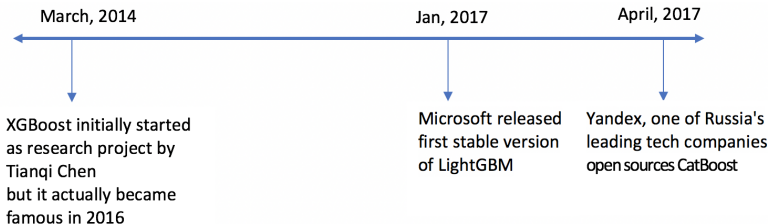
¹¹Ke, G. et al. (2017). "Lightgbm: A highly efficient gradient boosting decision tree".

¹²Prokhorenkova, L., et al. (2017). "CatBoost: unbiased boosting with categorical features".



Алгоритмы XGBoost, LightGBM и CatBoost

Наибольшую популярность на данный момент имеют реализации градиентного бустинга на решающих деревьях⁹:



Одни из наиболее известных на данный момент - это XGBoost¹⁰, LightGBM¹¹ и CatBoost¹².

⁹<https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db>

¹⁰Chen, T., and Guestrin, C. (2016). "Xgboost: A scalable tree boosting system".

¹¹Ke, G. et al. (2017). "Lightgbm: A highly efficient gradient boosting decision tree".

¹²Prokhorenkova, L., et al. (2017). "CatBoost: unbiased boosting with categorical features".



Алгоритмы XGBoost, LightGBM и CatBoost

Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,



Алгоритмы XGBoost, LightGBM и CatBoost

Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,
- Способом работы с категориальными признаками (например, XGBoost не умеет с ними работать),



Алгоритмы XGBoost, LightGBM и CatBoost

Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,
- Способом работы с категориальными признаками (например, XGBoost не умеет с ними работать),
- Скоростью обучения / тестирования.



Алгоритмы XGBoost, LightGBM и CatBoost

Эти реализации отличаются:

- Методом ветвления в узлах дерева при его обучении,
- Способом работы с категориальными признаками (например, XGBoost не умеет с ними работать),
- Скоростью обучения / тестирования.

	XGBoost	Light BGM		CatBoost	
Parameters Used	max_depth: 50 learning_rate: 0.16 min_child_weight: 1 n_estimators: 200	max_depth: 50 learning_rate: 0.1 num_leaves: 900 n_estimators: 300		depth: 10 learning_rate: 0.15 l2_leaf_reg= 9 iterations: 500 one_hot_max_size = 50	
Training AUC Score	0.999	Without passing indices of categorical features	Passing indices of categorical features	Without passing indices of categorical features	Passing indices of categorical features
		0.992	0.999	0.842	0.887
Test AUC Score	0.789	0.785	0.772	0.752	0.816
Training Time	970 secs	153 secs	326 secs	180 secs	390 secs
Prediction Time	184 secs	40 secs	156 secs	2 secs	14 secs
Parameter Tuning Time (for 81 fits, 200 iteration)	500 minutes	200 minutes		120 minutes	



- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,



- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),



- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,



- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,
- Бэггинг лучше всего параллелится,



- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,
- Бэггинг лучше всего параллелится,
- Бустинг позволяет фильтровать выбросы,

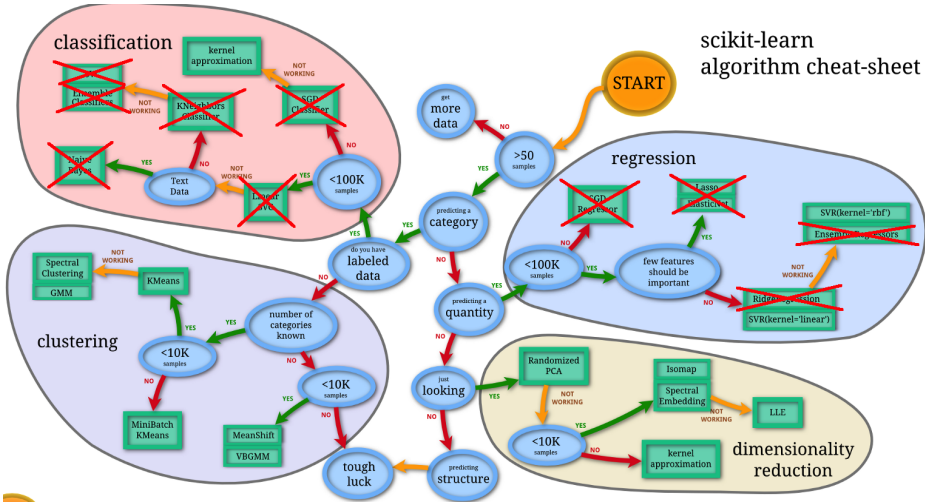


Сравнение алгоритмов композиции

- Бэггинг (благодаря процедуре бутстрэпа) может работать на небольших выборках,
- Бустинг лучше работает на больших выборках (но и ошибка, скорее всего, будет меньше),
- Стековое обобщение можно использовать как средство “выжимания” последних долей процента,
- Бэггинг лучше всего параллелится,
- Бустинг позволяет фильтровать выбросы,
- Метод случайных подпространств (бутстрэп на признаках) необходим, когда у нас признаков очень много (или много шумовых).



Дорожная карта Scikit-Learn¹³



¹³https://scikit-learn.org/stable/tutorial/machine_learning_map/

На основе материалов сайта <http://www.machinelearning.ru>.